Tarea DOM

Fichero XML

Guarda el siguiente XML como **coches.xml** en el mismo directorio desde donde ejecutes el programa (o ajusta la ruta en el parse):

```
<?xml version="1.0" encoding="UTF-8"?>
<grupo curso="2DAM" tutor="Jose Luis">
 <alumno id="A01">
  <nombre>Fran</nombre>
  <edad>22</edad>
  <modulos>
   <modulo>Acceso a Datos</modulo>
   <modulo>Desarrollo de Interfaces</modulo>
  </modulos>
 </alumno>
 <alumno id="A02">
  <nombre>Sonia</nombre>
  <edad>21</edad>
  <modulos>
   <modulo>PSP</modulo>
   <modulo>Acceso a Datos</modulo>
  </modulos>
 </alumno>
 <alumno id="A03">
  <nombre>Javier</nombre>
  <edad>23</edad>
  <modulos>
   <modulo>SI</modulo>
  </modulos>
 </alumno>
</grupo>
```

Requisitos

1. Lee el XML y muestra por consola
2. Para cada <alumno> añade un <email> si no existe.
   Regla: id en minúsculas (.toLowerCase( )) + "@digitech.es".
3. Marca con un atributo ad="true" el <alumno> que curse “Acceso a Datos”.
4. Guarda el resultado como alumnos_out.xml con indentación.

API DOM

• DocumentBuilderFactory.newInstance(), setNamespaceAware(true), newDocumentBuilder().parse(...)

• doc.getElementsByTagName("alumno") → NodeList

• Para cada Element alumno:

getAttribute("id"), getElementsByTagName("nombre").item(0).getTextContent()

• Crear nodos:

doc.createElement("email"), setTextContent(...), appendChild(...)

• Atributo:

alumno.setAttribute("ad","true")

• Guardar:

TransformerFactory.newInstance().newTransformer() + transform(new DOMSource(doc), new StreamResult(new File("alumnos_out.xml")))

• Indentación:

t.setOutputProperty(OutputKeys.INDENT, "yes")

Plantilla inicial (para completar)

```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.File;

public class EjercicioDOM {
    public static void main(String[] args) throws Exception {
        DocumentBuilderFactory f =
DocumentBuilderFactory.newInstance();
        f.setNamespaceAware(true);
        DocumentBuilder b = f.newDocumentBuilder();
        Document doc = b.parse(new File("alumnos.xml"));

        Element raiz = doc.getDocumentElement(); // <grupo>
        NodeList alumnos = doc.getElementsByTagName("alumno");

        // 1: imprimir "id: nombre (edad)"

        // 2: añadir <email> si no existe -> id en minúsculas +
"@digitech.es"

        // 3: si hay <modulo>Acceso a Datos</modulo> ->
setAttribute("ad","true")



        // Guardar
        Transformer t =
TransformerFactory.newInstance().newTransformer();
        t.setOutputProperty(OutputKeys.INDENT, "yes");
        t.transform(new DOMSource(doc), new StreamResult(new
File("alumnos_out.xml")));
    }
}
```

Tarea SAX

Implementar un **lector SAX** que procese un XML sencillo de coches, imprima los datos de cada <coche> por consola y muestre el **total de coches** al finalizar. El código base contiene huecos marcados como **RELLENAR** que debes completar.

Fichero XML

Guarda el siguiente XML como **coches.xml** en el mismo directorio desde donde ejecutes el programa (o ajusta la ruta en el parse):

```
<?xml version="1.0" encoding="UTF-8"?>
<coches>
 <coche>
  <marca>Seat</marca>
  <modelo>Ibiza</modelo>
  <color>rojo</color>
  <matriculacion>2019</matriculacion>
 </coche>
 <coche>
  <marca>Renault</marca>
  <modelo>Clio</modelo>
  <color>azul</color>
  <matriculacion>2021</matriculacion>
 </coche>
</coches>
```

Requisitos

1. Al encontrar <marca>, <modelo>, <color> o <matriculacion>, imprime su valor en consola con el formato:

   Marca: ...

   Modelo: ...

   Color: ...

   Matriculacion: ...

2. Cada vez que termine un </coche>, incrementa el **contador**.
3. Al finalizar el documento, muestra:

   Total: N coche(s)
4. Usa el **patrón de flags (booleanos)** mostrado en la plantilla para saber qué etiqueta se está procesando.
5. Recuerda que characters(...) puede llamarse varias veces; usa trim() y desactiva el flag tras imprimir.

Plantilla (para completar) MainSaxCoches.java

```
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

public class MainSaxCoches {
    public static void main(String[] args) {
        try {
            // Factoría y parser SAX
            SAXParserFactory factory = /* RELLENAR */;
            SAXParser saxParser = /* RELLENAR */;

            // Handler
            SaxHelper handler = /* RELLENAR */;

            // Parseo (ajusta la ruta a tu proyecto)
            saxParser.parse(/* RELLENAR */, handler);

        } catch (/* RELLENAR */ e) {
            e.printStackTrace();
        }
    }
}
```

Plantilla (para completar) SaxHelper.java

```
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class SaxHelper extends DefaultHandler {

    // Banderas
    boolean esMarca = false;
    boolean esModelo = false;
    boolean esColor = false;
    boolean esMatriculacion = false;

    // Contador de coches
    int totalCoches = /* RELLENAR */;

    @Override
    public void startElement(String uri, String localName, String
elementos, Attributes atributos)
            throws SAXException {

        System.out.println(/* RELLENAR: */);

        switch (/* RELLENAR */) {
            case /* RELLENAR */:
                break;
             /* RELLENAR */
                    …
            default:
                break;
        }
    }
```

```
    @Override
    public void characters(char[] ch, int inicio, int length) throws
SAXException {
        // El texto puede venir troceado; usamos trim() para evitar
líneas vacías
        String texto = /* RELLENAR con .trim()al final */;
        if (/* RELLENAR: texto.isEmpty() */) return;

        if (/* RELLENAR */) {
            System.out.println(/* RELLENAR */);
           /* RELLENAR */= false;
            return;
        }

        /* RELLENAR */
                    …
        }
    }

    @Override
    public void endElement(String uri, String localName, String
elementos) throws SAXException {
        System.out.println(/* RELLENAR */);

        if (/* RELLENAR: "coche".equals(elementos) */) {
            // al cerrar un <coche> sumamos 1
            totalCoches++;
        }
    }

    @Override
    public void endDocument() throws SAXException {
        System.out.println(/* RELLENAR */);
    }
}
```